UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/591,317	08/31/2006	Evgeny Kazakov	P-6625-US	9428

49443 7590 12/24/2009

Pearl Cohen Zedek Latzer, LLP
1500 Broadway
12th Floor
New York, NY 10036

EXAMINER	
CHOW, CHIH CHING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
12/24/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/591,317	KAZAKOV ET AL.	
	Examiner	Art Unit	
	CHIH-CHING CHOW	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *31 August 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-39* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-39* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *31 August 2006* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *1/29/09*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed on August 31, 2006.

2. The priority date considered for this application is March 02, 2004, which is the filing date of the provisional application no. 60/548,879.

3. Claims 1-39 have been examined.

Specification

4. The use of the trademark JAVA has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks. To expedite correction on this matter, the examiner suggests the following guidelines for Applicant to follow in amending the specification:

a. Capitalize each letter of a trademark or accompany the trademark with an appropriate designation symbol, e.g., ™ or ®, as appropriate.

b. Use each trademark as an adjective modifying a descriptive noun. For example, it would be appropriate to recite "a JAVA class file" or "a JAVA application". Note that in these examples, "class file" and "application" provide accompanying generic terminology, describing the context in which the trademark is used. By itself, the trademark JAVA specifies only the source of the so-labeled products, namely SUN Microsystems, Inc.

Claim Rejections - 35 USC § 112

5. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

6. Claim 12 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 12 recites "The method of claim 9, wherein defining said domain-specific language further comprises: defining an aspect able to affect an element selected from a group consisting of: said at least one language term, a property of said at least one language term, and a relationship between said at least one language term and another language term." The only related description is in paragraph [0029], "The method may include, for example, defining an aspect able to affect an element selected from a group consisting of: the at least one language term, a property of the at least one language term, and a relationship between the at least one language term and another language term." – it's not clear to the Examiner, what does 'an aspect able to affect an element from a group' mean? Is it mean defining an attribute for a certain element from a group? Such as a text's (element) color and font, which is selected from a group (e.g. displaying property-rules)?

7. Claim 15 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 15 recites "The method of claim 14, wherein creating comprises: generating a recommended modeling route to be used during creation of said one or more elements of said model in accordance with a mentor modeling definition of said domain-specific language." The only

related description is in paragraph [0032], "The method may include, for example, generating a recommended modeling route to be used during creation of the one or more elements of the model in accordance with a mentor modeling definition of the domain-specific language." – it's not clear to the Examiner, what does 'a recommended modeling route' and 'a mentor modeling definition' mean? Is it mean a design pattern or a client module's context?

8. Claim 27 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 27 recites "The system of claim 19, comprising a mentoring module to generate a recommended modeling route available during creation of one or more elements of a model in accordance with a mentor modeling definition of said domain-specific language." The only related description is in paragraph [0032], "The method may include, for example, generating a recommended modeling route to be used during creation of the one or more elements of the model in accordance with a mentor modeling definition of the domain-specific language." – it's not clear to the Examiner, what does 'a recommended modeling route' and 'a mentor modeling definition' mean? Is it mean a design pattern or a client module's context?

9. Claim 32 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 32 recites "The system of claim 29, wherein said language runtime module comprises a process mentor module to guide said runtime process in accordance with a process definition of said domain-specific language." The only related description is in paragraph [0040], "In some embodiments, for example, the language runtime module may include a process

mentor module to guide the runtime process in accordance with a process definition of the domain-specific language." – it's not clear to the Examiner, what does 'process mentor module to guide said runtime process' mean? Is it mean a design pattern or a client module's context?

Claim Rejections – 35 USC § 101

10. 35 U.S.C. § 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the condition and requirements of this title.

11. Claim 1 is rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter. A § 101 process must (1) be tied to another statutory class (a particular machine or apparatus) or (2) transform underlying subject matter (such as an article or materials) to a different state or thing. On this basis, claim rejected under 35 U.S.C. § 101.

12. Regarding claims 2-18, each of these claims is rejected based on its dependency to independent claim 1, and do not cure the deficiency of claim 1.

13. Claim 19 is rejected under 35 U.S.C. § 101 because the claimed invention is directed to non-statutory subject matter.

Claim 19 merely recites a language builder module, a dynamic component, a static component, i.e., computer program per se. On this basis, claim rejected under 35 U.S.C. § 101.

14. Regarding claims 20-35, each of these claims is rejected based on its dependency to independent claim 19, and do not cure the deficiency of claim 19.

Claim Rejections - 35 USC § 102

15. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

16. Claims 1-6, 8-21, 23-32, 34-39 are rejected under 35 U.S.C. 102(b) as being anticipated by US Patent No. 6,269,473 by Freed et al., hereinafter "Freed".

As Per claim 1, Freed discloses:

- ***A method comprising: defining a domain-specific language usable in a modeling environment and having a dynamic component and a static component, said dynamic component able to affect a behavior of said static component.***

See Freed's FIG. 26, and description in column 19, lines 46-51, "the DCT 26.6, is implemented 26.14 in a TSE 26.4 using data structures and algorithms 26.8 *(wherein the data structures and the algorithms are considered as static components)*, Similarly the DCO 26.10 *(the Dynamically Configurable Object is considered as dynamic components)*, is implemented 26.16 in a TSE 26.4 using data structures and algorithms 26.12. It is a simplification but is not necessary that a single type or instance of a TSE 26.4 is used for all of the DCTs 26.6 *(the Dynamically Configurable Types are considered as dynamic components)* or DCOs 26.10 as long as an appropriate communication is supported between **the TSEs 26.4**." Further, lines 60-65, "The DCT 26.6 is preferably directly implemented using standard class or type definitions within a TSE 26.4 but

may be abstracted above these constructs. For instance in Smalltalk, C++ or **Java TSEs** 26.4 these could be directly based on standard class or type constructs as provided in the associated **TSE 26.4 language**" – item 26.4 is the **Target Software Environment language**, which is a domain-specific language. Freed's teaching uses Target Software Environment to define a specific domain of application, and it contains both dynamic components (dynamically configurable objects and types) and static components (data structures and algorithms); wherein the dynamically configurable objects and types can effect the data structures and the algorithms, see column 11, lines 4-8, "The abstraction of a module is commonly considered its type and defines a public interface that is a definition of services that it can provide to other modules. **A configurable module can keep this type invariant upon configuration or allow it to change**. If it is allowed to change, **then the dynamically configurable model supports dynamic-typing**."; column 10, lines 32-41, "**Dynamism is defined as the expression of a software solution in a form that can change at run-time**. This means that the module should be able to **express configurability while the software is being executed**. This is analogous to an individual who can change behaviors instantly as necessary in response to dynamically changing demands of the individual, or individuals and groups interacting with that individual. This allows configuration, as described above, to be applied not only during design-time but also during runtime." And column 8 last line to column 9, line 3, "This is in order to **allow the defined objects to have different semantics over time to serve different purposes**, the end result

of which is that a software solution based on the **defined objects allows dynamic configurability**".

As Per claim 2, Freed discloses:

- ***The method of claim 1, wherein defining said domain-specific language comprises: defining said dynamic component and said static component in accordance with Unified Modeling Language constructs and semantics.***

Claim 1 rejection is incorporated, further see Freed's Fig. 28-30, 32-62, 65-68 for UML constructs and semantics.

As Per claim 3, Freed discloses:

- ***The method of claim 1, wherein defining said domain-specific language comprises: defining a customized Unified Modeling Language meta-modeling profile which supports definitions of said dynamic component and said static component.***

Claim 1 rejection is incorporated, further see claim 2 rejection, all the UML listed in claim 2 rejection defines a customized UML meta-modeling profile; also see column 19, lines 41-45, "The DCT 25.8, is a specification that is instantiated 25.14, to one or more instances of run-time constructs 25.6 called a Dynamic Configurable Object (DCO) 25.10. The DCT 25.8, represents the design-time **meta-data** *(meta-modeling profile) f*or the DCSM 25.2, version of a dynamically configurable module with configurable implementation. The DCO 25.10, is the run-time embodiment of a dynamically configurable module with configurable implementation."

As Per claim 4, Freed discloses:

- ***The method of claim 1, wherein defining said domain-specific language comprises: defining said domain-specific language based on custom meta-modeling constructs, said constructs in accordance with a Unified Modeling Language meta-modeling profile and defining said dynamic component and said static component.***

Claim 1 rejection is incorporated, further FIG. 29 is a **UML** depiction of the DCTs (Dynamic Configurable Types) according to a preferred embodiment of the present invention. – UML meta-modeling for dynamic components; FIG. 32 is a UML depiction of the types of properties according to a preferred embodiment of the present invention, wherein the property types are considered as a 'data structure', which is a static component; also FIG. 35 is a UML depiction of the rule-property according to a preferred embodiment of the present invention, which is also a static component.

As Per claim 5, Freed discloses:

The method of claim 1, wherein defining said domain-specific language comprises: importing a definition of an element of said domain-specific language from a previously-defined domain-specific language.

Claim 1 rejection is incorporated, further see column 15, lines 18-30, "FIG. 15, the two use-cases 15.6, 15.8, are handled by configurations 15.2, 15.4, encapsulated inside module 15.10. Extending this to the module-graph level, one can see that fine and coarse grained process-modeling is performed with higher modularity and abstraction, leading to higher simplicity and **reuse**. In

order to solidify this applicability of configurable module-graphs to process-modeling, the DCSM introduces two new concepts, the dynamically configurable process (DCP) and the dynamically configurable subprocess (DCS). A DCP is defined as a context for a client, either a human or an external system, to interact with the semantics of the system to perform a particular task." – previously defined UML can be reused (imported from external system), for UML feature see claims 2, 3 and 4 rejections.

As Per claim 6, Freed discloses:

The method of claim 1, wherein defining said domain-specific language comprises: validating said domain-specific language in accordance with a validation rule defined in a meta-modeling language.

Claim 1 rejection is incorporated, further see FIG. 46, which is a UML depiction of the **validation feature** according to a preferred embodiment of the present invention; also see description in column 26, lines 62-66, "As shown in FIG. 46, the **validate-feature 46.6 has a generalization relationship 46.4 to the type feature 46.2 and includes an attribute validation** 46.8 of type string **contains a specification in string form of how the associated property's visualization should be validated** or constrained on the client as the user edits it." – validating domain-specific language according to validation rule in a meta-modeling language (*UML*).

As Per claim 8, Freed discloses:

The method of claim 1, wherein defining said domain-specific language comprises: defining a custom action available for execution, on an

element of an application model compliant with said domain-specific language, in response to an invocation request in accordance with said domain-specific language.

Claim 1 rejection is incorporated, further see column 15, lines 40-45, "A DCS (*Dynamically Configurable Subprocess)*, like a standard function, has a signature, with associated input parameters or domain, and output parameters or range. The subprocess is intended to act as a function, mapping domain inputs to range outputs. Because of this, **a subprocess can perform a function call to another subprocess**." – a DCS is a defined custom action available for execution, which is an element of the domain-specific language defined application model.

As Per claim 9, Freed discloses:

- ***The method of claim 1, wherein defining said domain-specific language comprises: defining at least one language information item of said domain-specific language; defining at least one language term of said domain-specific language; and defining at least one data type of said domain-specific language.***

Claim 1 rejection is incorporated, FIG. 36 is a UML depiction of the property according to a preferred embodiment of the present invention, wherein the property definition contains language information (see 36.16, types, void, parameters…etc.); further FIG. 34 is a UML depiction of the method-property according to a preferred embodiment of the present invention, wherein the method-property is considered as a language term; FIG. 32 is a UML depiction of the types of properties according to a

preferred embodiment of the present invention, wherein the property types are considered as a data type.

As Per claim 10, Freed discloses:

- ***The method of claim 9, wherein defining said domain-specific language further comprises: defining a relationship between said at least one language term and another language term of said domain-specific language.***

Claim 9 rejection is incorporated, see FIG. 33 is a UML depiction of the **relationship-property** according to a preferred embodiment of the present invention.

As Per claim 11, Freed discloses:

- ***The method of claim 9, wherein defining said domain-specific language further comprises: defining a constraint associated with one or more elements of said domain-specific language to be used during validation of said one or more elements of said domain-specific language.***

Claim 9 rejection is incorporated, further FIG. 35 is a UML depiction of the **rule-property** (*constraint*) according to a preferred embodiment of the present invention.

As Per claim 12, Freed discloses:

- ***The method of claim 9, wherein defining said domain-specific language further comprises: defining an aspect able to affect an element selected from a group consisting of: said at least one language term, a property of***

said at least one language term, and a relationship between said at least one language term and another language term.

Claim 9 rejection is incorporated, further FIG. 43 is a UML depiction of the color feature according to a preferred embodiment of the present invention;

and FIG. 44 is a UML depiction of the font feature according to a preferred embodiment of the present invention. – both color and font are an aspect able to affect an element selected from a group.

As Per claim 13, Freed discloses:

- ***The method of claim 1, further comprising: applying said domain-specific language to said model during execution of a modeling process of said model.***

Claim 1 rejection is incorporated, further see column 15, lines 32-35, "Dynamism is defined as the expression of a software solution in a form that **can change at run-time**. This means that the module should be able **to express configurability while the software is being executed**." – using the domain-specific language model during execution of a modeling process.

As Per claim 14, Freed discloses:

- ***The method of claim 13, further comprising: creating one or more elements of a model in accordance with at least one language term defined in said domain-specific language.***

Claim 13 rejection is incorporated, further see claim 9 rejection.

As Per claim 15, Freed discloses:

- ***The method of claim 14, wherein creating comprises: generating a recommended modeling route to be used during creation of said one or more elements of said model in accordance with a mentor modeling definition of said domain-specific language.***

Claim 14 rejection is incorporated, further see column 17, lines 53-62, "Each form of configuration, along with each configuration **design pattern** that a DCSM user chooses to use to model an aspect of a configurable solution, can be considered a dimension in this multi-dimensional space. For instance, the designer may want to use configuration for a compensation calculation along the dimension of a continuum of skill level for an employee module, and also along the dimension of seniority. Each of these is **dependent on the client module's context,** and the current state of the server module, and the choices made at the policy layer".

As Per claim 16, Freed discloses:

- ***The method of claim 14, wherein creating comprises: executing a custom action defined in said domain-specific language on at least one of said one or more elements of said model.***

Claim 14 rejection is incorporated, see column 18, lines 52-55, "Since the DCSM supports encapsulated context-dependent semantics, it includes a declarative method of **defining the graphical user-interface (GUI) semantics**, both in terms of visualization and event-handling. This is done in such a way that it can allow configuration to drive the automatic generation of a GUI at run-time" -- a custom action.

As Per claim 17, Freed discloses:

- ***The method of claim 14, wherein creating comprises: converting a domain-specific model artifact of said domain-specific language into an application artifact usable during execution of said modeling process.***

Claim 14 rejection is incorporated, further see column 13, lines 25-34, "A well defined abstraction contains little of the non-essentials of a module's service definition. **By placing context-dependent semantics into the server-module, the artifacts of these semantics do not show up in the abstraction**. For example, if a module is not responsible for its own event management, then the abstraction of the module, would potentially have to include **passing of an event management context from its client module** (*application artifact)*. This would mean including non-essential implementation details in what should be as pure a form of abstraction as possible."

As Per claim 18, Freed discloses:

- ***The method of claim 17, further comprising: storing said domain-specific model artifact in a metadata database able to provide access to said domain-specific model artifact.***

Claim 17 rejection is incorporated, further see column 12, lines 43-49, "Context-independent semantics are defined to be those semantics of a module that do not change no matter what module it has as a client, and no matter how that client module intends to use the services of that particular module. Context-independent semantics include State (relationships and attributes), Behavior (methods) and Persistence (**database storage**)."

As Per claim 19, Freed discloses:

- ***A system for accelerated modeling, the system comprising: a language builder module to define a domain-specific language usable in a modeling environment and having a dynamic component and a static component, said dynamic component able to affect a behavior of said static component.***

Claim 19 is the system version of claim 1, see claim 1 rejection.

As Per claim 20, Freed discloses:

- ***The system of claim 19, wherein said language builder module is able to import a definition of an element of said domain-specific language from a previously-defined domain-specific language.***

Claim 19 rejection is incorporated, claim 20 is the system claim version for claim 5, therefore see claim 5 rejection.

As Per claim 21, Freed discloses:

- ***The system of claim 19, wherein said language builder module comprises a validator to validate said domain-specific language in accordance with a validation rule defined in a meta-modeling language.***

Claim 19 rejection is incorporated, claim 21 is the system claim version for claim 6, therefore see claim 6 rejection.

As Per claim 23, Freed discloses:

- ***The system of claim 19, wherein said language builder module comprises an action editor to define a custom action available for***

execution on a model in accordance with said domain-specific language in response to an invocation request in accordance with said domain-specific language.

Claim 19 rejection is incorporated, claim 23 is the system claim version for claim 8, therefore see claim 8 rejection.

As Per claim 24, Freed discloses:

- ***The system of claim 19, wherein said language builder module is able to define at least one language information item of said domain-specific language, to define at least one language term of said domain-specific language, and to define at least one data type of said domain-specific language.***

Claim 19 rejection is incorporated, claim 24 is the system claim version for claim 9, therefore see claim 9 rejection.

As Per claim 25, Freed discloses:

- ***The system of claim 24, wherein said language builder module is able to define a relationship between said at least one language term and another language term of said domain-specific language.***

Claim 24 rejection is incorporated, claim 25 is the system claim version for claim 10, therefore see claim 10 rejection.

As Per claim 26, Freed discloses:

- ***The system of claim 19, wherein said language builder module comprises a constraint editor to define a constraint associated with said***

data type to be used during validation of one or more elements of said domain-specific language.

Claim 19 rejection is incorporated, claim 26 is the system claim version for claim 11, therefore see claim 11 rejection.

As Per claim 27, Freed discloses:

- The system of claim 19, comprising a mentoring module to generate a recommended modeling route available during creation of one or more elements of a model in accordance with a mentor modeling definition of said domain-specific language.

Claim 19 rejection is incorporated, claim 27 is the system claim version for claim 15, therefore see claim 15 rejection.

As Per claim 28, Freed discloses:

- The system of claim 19, comprising a generator able to create one or more elements of a model in accordance with a process defined in said domain-specific language.

Claim 19 rejection is incorporated, further see column 32, lines 37-48, "FIG. 70 illustrates the integrated strategy where integrated design-time constructs 70.2, which are TSE representations of DCSM design-time constructs along with DCSM annotations are shown. These are read 70.8 by a scanner 70.4 that can read the TSE specific model and extract the DCSM design-time information. The results of this scan are passed 70.12 to **a generator 70.6**, which **regenerates 70.10 the integrated design-time constructs** 70.2, *(one or more elements of a model)* preserving the DCSM design-time

specification. The combination of the scanner 70.4, and the generator 70.6, make up a DCSM development tool 70.14 that is used in conjunction with the native development tools of the TSE to create a DCSS."

As Per claim 29, Freed discloses:

- ***The system of claim 28, comprising a language runtime module to apply said domain-specific language to said model during execution of a runtime process of said model.***

Claim 28 rejection is incorporated, further see column 10, lines 32-41, "**Dynamism is defined as the expression of a software solution in a form that can change at run-time**. This means that the module should be able to **express configurability while the software is being executed**. This is analogous to an individual who can change behaviors instantly as necessary in response to dynamically changing demands of the individual, or individuals and groups interacting with that individual. This allows configuration, as described above, to be applied not only during design-time but also during runtime."

As Per claim 30, Freed discloses:

- ***The system of claim 29, wherein said language runtime module comprises a validator to validate said model based on a validation rule defined in said domain-specific language.***

Claim 29 rejection is incorporated, claim 30 is the system claim version for claim 6, therefore see claim 6 rejection.

As Per claim 31, Freed discloses:

- ***The system of claim 29 wherein said language runtime module comprises an action executor to execute a custom action defined in said domain-specific language on at least one of said one or more elements of said model.***

Claim 29 rejection is incorporated, claim 31 is the system claim version for claim 8, therefore see claim 8 rejection.

As Per claim 32, Freed discloses:

- ***The system of claim 29, wherein said language runtime module comprises a process mentor module to guide said runtime process in accordance with a process definition of said domain-specific language.***

Claim 29 rejection is incorporated, claim 32 is the system claim version for claim 15, therefore see claim 15 rejection.

As Per claim 34, Freed discloses:

- ***The system of claim 29, comprising a converter to convert a domain-specific model artifact based on said domain-specific language into an application artifact usable during execution of said runtime process.***

Claim 29 rejection is incorporated, claim 34 is the system claim version for claim 17, therefore see claim 17 rejection.

As Per claim 35, Freed discloses:

- ***The system of claim 34, comprising a database to store said domain-specific model artifact and to provide access to said domain-specific model artifact during execution of said runtime process.***

Claim 34 rejection is incorporated, claim 35 is the system claim version for claim 18, therefore see claim 18 rejection.

As Per claim 36, Freed discloses:

- ***A machine-readable medium having stored thereon instructions that, when executed by a machine, result in: defining a domain-specific language usable in a modeling environment and having a dynamic component and a static component, said dynamic component able to affect a behavior of said static component.***

Claim 36 is the machine-readable medium version of claim 1, see claim 1 rejection.

As Per claim 37, Freed discloses:

- ***The machine-readable medium of claim 36, wherein the instructions result in: defining said dynamic component and said static component in accordance with Unified Modeling Language constructs and semantics.***

Claim 36 rejection is incorporated, claim 37 is the machine-readable medium version for claim 2, therefore see claim 2 rejection.

As Per claim 38, Freed discloses:

- ***The machine-readable medium of claim 36, wherein the instructions result in: defining a customized Unified Modeling Language meta-***

modeling profile which supports definitions of said dynamic component and said static component.

Claim 36 rejection is incorporated, claim 38 is the machine-readable medium version for claim 3, therefore see claim 3 rejection.

As Per claim 39, Freed discloses:

- The machine-readable medium of claim 36, wherein the instructions result in: defining said domain-specific language based on custom meta-modeling constructs, said constructs in accordance with a Unified Modeling Language meta-modeling profile and defining said dynamic component and said static component.

Claim 36 rejection is incorporated, claim 39 is the machine-readable medium version for claim 4, therefore see claim 4 rejection.

Claim Rejections - 35 USC § 103

17. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

18. Claims 7, 22, and 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,269,473 by Freed et al., hereinafter "Freed".

As Per claim 7,

- ***The method of claim 1, wherein defining said domain-specific language comprises: generating an eXtensible Markup Language output representing at least one definition of said domain-specific language.***

Claim 1 rejection is incorporated, Freer teaches a domain-specific language that is usable in a modeling environment, having a dynamic component and a static component, but he does not teach 'generating an eXtensible Markup Language (XML) output' specifically, however, XML is well known to the people in the art to facilitate sharing of structured text and information across the Internet.

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Freed's disclosure of a domain-specific language that is usable in a modeling environment, having a dynamic component and a static component, by using XML to output the definition of the domain-specific language. The modification would be obvious because one of ordinary skill in the art would be motivated to share the domain-specific language defined model across the Internet.

As Per claim 22,

- ***The system of claim 19, wherein said language builder module comprises a generator to generate an eXtensible Markup Language output representing at least one definition of said domain-specific language.***

Claim 19 rejection is incorporated, claim 22 is system version of claim 7, therefore see claim 7 rejection.

As Per claim 33,

- ***The system of claim 29, wherein said language runtime module comprises a generator to generate an extensible Markup Language output representing said model based on said domain-specific language.***

Claim 29 rejection is incorporated, further see claim 7 rejection.

Conclusion

19. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Makhlouf, US Patent No. 6,789,054, discloses an invention that specifies physics-like computational models called M models. These models enable the specification and design automation of an integrated architecture of a real system. A new class of adaptive systems engineering tools called Adaptive Model-Reference (AMR) tools, supports the M computational model. AMR tools are used to visually specify the integrated architecture model of a real system, assess it's value, and automate the static and dynamic binding of it's model components into adaptive systems. These systems can be adapted during system development to compensate for requirement changes and design errors, and during run time operation to compensate for unanticipated operational system conditions. AMR tools enable the verification and validation of adaptive real system designs built in compliance with a declared enterprise wide technical architecture. Architecture components can be specified using AMR tools or can be imported into the AMR tool set. AMR tools are specified using an open system architecture built as extensions to open system development environments.

Coulthard et al., US 2002/0178290, discloses a method and apparatus that has the ability to convert display pages of interactive legacy applications for use on a network, such as the Internet. The display source code of the legacy application

is first parsed into a network-based language, such as XML, preserving the structure and hierarchy of the display source to create a plurality of network user interface pages. The network user interface pages are then converted to a dynamic platform-independent language in which the static portion of the display page is converted to a web page, such as a JavaServer Page, and the dynamic portion of the display page for input/output/feedback is converted to data objects, such as JavaBeans. The intermediate network user interface pages may be stored on the server with the legacy application.

Ben-Romdhane, US 2004/0031015, discloses a system and method for manipulating software is provided. The system analyzes a body of source code and derives a set of components from the software architecture inherent in the body of source code. The system further determines the internal and external functional and data dependencies of each component and the control flow of the software and creates an information model that represents the components, their dependencies, and the control flow of the program.

Konstantinou et al., US 2005/0097146, discloses methods and systems for autonomously managing computer networks are provided. In a preferred embodiment of the invention, management functions are organized in a novel two-layer peer-to-peer (P2P) architecture. The bottom layer organizes management information in a unified object-relationship model, that is instantiated in a distributed transactional object repository. The top layer unifies the traditional roles of managers and elements into a single autonomic management peering layer. Autonomic elements use the repository as a primary management repository, and effect autonomic behavior in terms of transactions over the shared model state.

Hunt, US Patent No. 7,493,630, discloses a computer readable medium

contains a data structure, which, in some embodiments, is an interface wrapper. The interface wrapper stores information about the interface such as a first data field that contains a reference to instrumentation, a second data field that contains data which represents a reference to an interface of a software program, and a third data field which contains data representing an identity of the unit of software. When a client unit calls a member function of the interface, the interface wrapper intercepts the call and invokes the instrumentation.

Katzeff, US Patent No. 5,101,491, discloses a Synthesizer means for generating software for a computer which is programmed for controlling a physical system. The software generated by the synthesizer represents a new function to be incorporated in the existing system. The synthesizer includes a device for receiving a formal description representative of the new function in a specification language and for translating the specification into a base document. The base document is further processed by document processing devices for handling the static, interface and dynamic parts of the description to produce an error-free base document. The complete base document is translated by an information processing device into an internal code document which is used by a check device and a simulation device.

Toutonghi et al., US Patent No. 6,438,744, discloses the dynamic mapping from an ActiveX component model to a JavaBean model is disclosed. In one embodiment, an ActiveX compatible object is created at run-time for those JavaBean components that an ActiveX client application wishes to utilize. In another embodiment, upon a call to CoGetClassObject or related methods using class identifiers, a COM-callable wrapper is generated and a mapping table containing COM dispatch identifiers is scanned to ensure the dispatch identifiers are unique.

Hanna et al., US Patent No. 5,748,961, discloses a software system is defined by a tree of system models which are written in a functional language. During a build of the software system, the functions are interpreted and the results of the expensive expressions are cached. Each function is examined before interpretation to see if it has been evaluated before. If a function has already been evaluated, the cached result is retrieved by the evaluator and the time which would have been spent re-evaluating the function is saved.

Smith et al., US Patent 7,627,861, discloses methods, systems, and computer program products for identifying computer source code constructs are disclosed. According to one method, computer source code is converted to a format suitable for an automated inference engine. The automated inference engine receives as inputs the converted source code, a set of elemental design patterns defining patterns to be identified, and a set of rules defining relationships between patterns. The automated inference engine outputs proofs indicative of patterns present in the source code. The proofs may be converted to a source code pattern report.

Bhargava et al., US patent 7,568,019, discloses a method for monitoring, predicting performance of, and managing Business Operations by the simultaneous, real-time Integration, Normalization and Correlation of direct measurements at the Business Layer and other Layers of Business Operations. The other Layers considered, may include, for example, Application and Infrastructure Layers. The system enables the user to automate sophisticated management tasks by Correlating measurements of activity, performance and availability at all Layers of Business Operations. Significantly, the techniques described herein extend the domain of the Correlations across real-time measurements from all Layers of

Business Operations, giving central importance to the measurements in the Business Layer within the Correlations.

Booch et al., "Unified Modeling Language for Object-Oriented Development", August 1996, discloses UML semantics, syntaxes, makes the metamodel cleaner and smaller, Etc.

20. The following summarizes the status of the claims:

35 USC § 112 rejection: Claims 12, 15, 27, 32

35 USC § 101 rejection: Claims 1-35

35 USC § 102 rejection: Claims 1-6, 8-21, 23-32, 34-38

35 USC § 103 rejection: Claims 7, 22, 33

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 8:30am - 5:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-

direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Chih-Ching Chow/
Examiner, Art Unit 2191
12/18/09

/Ted T. Vo/
Primary Examiner, Art Unit 2191